

DIALOG(R)File 351:Derwent WPI
(c) 2004 Thomson Derwent. All rts. reserv.

014202451 **Image available**

WPI Acc No: 2002-023148/ 200203

XRPX Acc No: N02-018519

Parallel program generation method in compiler, involves inserting data and loop dispersion indication statements for each detected array of a variable in mid level language read out from sequential program

Patent Assignee: HITACHI LTD (HITA)

Number of Countries: 001 Number of Patents: 001

Patent Family:

Patent No	Kind	Date	Applicat No	Kind	Date	Week
JP 2001297068	A	20011026	JP 2000109550	A	20000411	200203 B

Priority Applications (No Type Date): JP 2000109550 A 20000411

Patent Details:

Patent No	Kind	Lan Pg	Main IPC	Filing Notes
JP 2001297068	A	10	G06F-015/16	

Abstract (Basic): **JP 2001297068 A**

NOVELTY - The array of a variable is detected using a detector (5a) in a compiler (2). The data and loop dispersion indication statements are generated for each detected array and inserted in the mid level language read out from input sequential program (1). The parallel program (3) is generated by multithreading the nested loop containing parallel loops in the mid level language.

DETAILED DESCRIPTION - An INDEPENDENT CLAIM is also included for recorded medium storing instruction for parallel program generation.

USE - Used in compilers for generating parallel program for distributed shared memory type calculators.

ADVANTAGE - Generates optimum data dispersion by reducing the difference between data dispersion and loop dispersion, and thereby improving the processing velocity of the parallel program.

DESCRIPTION OF DRAWING(S) - The figure shows the block diagram of parallel program generation system. (Drawing includes non-English language text).

Input sequential program (1)

Compiler (2)

Parallel program (3)

Detector (5a)

pp; 10 DwgNo 1/13

Title Terms: PARALLEL; PROGRAM; GENERATE; METHOD; COMPILE; INSERT; DATA; LOOP; DISPERSE; INDICATE; STATEMENT; DETECT; ARRAY; VARIABLE; MID; LEVEL; LANGUAGE; READ; SEQUENCE; PROGRAM

Derwent Class: T01

International Patent Class (Main): G06F-015/16

International Patent Class (Additional): G06F-009/45

File Segment: EPI

Manual Codes (EPI/S-X): T01-F05A; T01-M02C

THIS PAGE BLANK (USPTO)

(19) 日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11) 特許出願公開番号
特開2001-297068
(P2001-297068A)

(43) 公開日 平成13年10月26日 (2001. 10. 26)

(51) Int.Cl. ⁷	識別記号	F I	テーマコード [*] (参考)
G 0 6 F 15/16 9/45	6 3 0	G 0 6 F 15/16 9/44	6 3 0 A 5 B 0 4 5 3 2 2 G 5 B 0 8 1

審査請求 未請求 請求項の数 2 O L (全 10 頁)

(21) 出願番号 特願2000-109550 (P2000-109550)

(22) 出願日 平成12年4月11日 (2000. 4. 11)

(出願人による申告) 産業再生法第30条の規定による特定研究成果に係る特許を受けようとする出願

(71) 出願人 000005108

株式会社日立製作所

東京都千代田区神田駿河台四丁目6番地

(72) 発明者 廣岡 孝志

神奈川県川崎市麻生区王禅寺1099番地 株

式会社日立製作所システム開発研究所内

(72) 発明者 太田 寛

神奈川県川崎市麻生区王禅寺1099番地 株

式会社日立製作所システム開発研究所内

(74) 代理人 100077274

弁理士 磯村 雅俊 (外1名)

Fターム (参考) 5B045 GG11

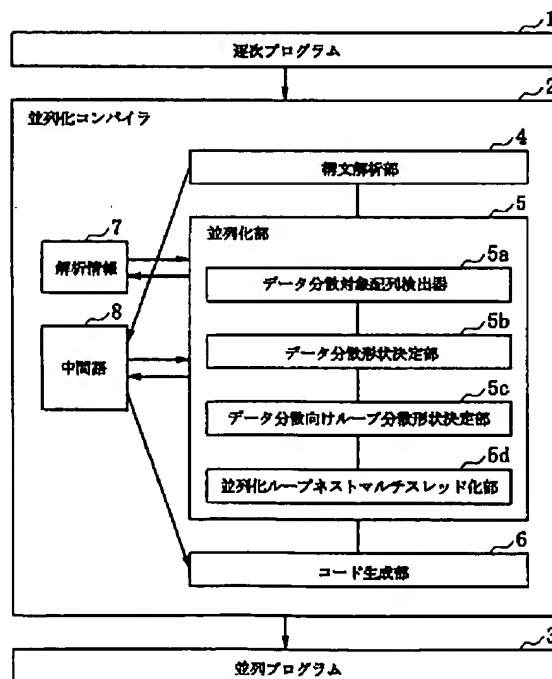
5B081 CC11 CC16 CC32

(54) 【発明の名称】 並列プログラム生成方法およびその記録媒体

(57) 【要約】

【課題】 分散共有メモリ型並列計算機において、最適なデータ分散を実現する並列プログラムを生成し、並列プログラムの処理速度を向上させる。

【解決手段】 並列化コンパイラ2の並列化部5において、まず、データ分散対象配列検出部5aにより、入力された逐次プログラム1から、ループ繰り返し範囲が変数のループ中に参照を有する配列、もしくは配列宣言寸法が変数の配列、もしくは引数配列を検出し、次に、データ分散形状決定部5bにより、この配列をページサイズにブロックサイクリック分散させるデータ分散指示文を生成して挿入し、さらに、データ分散向けループ分散形状決定部5cにより、このデータ分散形状と一致するループ分散形状となるループ分散指示文を生成して挿入し、そして、並列化ループネストマルチスレッド化部5dにより、並列化ループを含むネストループをマルチスレッド化することにより、並列プログラム3を生成する。



【特許請求の範囲】

【請求項1】 逐次実行用プログラムを入力してコンピュータ処理により分散共有メモリ型並列計算機用の並列プログラムに変換するコンパイル装置の並列プログラム生成方法であって、上記逐次実行用プログラムを入力して構文解析を行い中間語を記憶装置に出力するステップと、上記記憶装置から上記中間語を読み出して、ループ繰り返し範囲が変数のループ中に参照を有する配列、もしくは配列宣言寸法が変数の配列、もしくは引数配列を検出するステップと、検出した各配列をページサイズにブロックサイクリック分散させるデータ分散指示文を生成して上記中間語に挿入するステップと、上記データ分散指示文により分散される上記各配列のデータ分散形状と一致するループ分散形状となるループ分散指示文を生成して上記中間語に挿入するステップと、上記中間語における並列化ループを含むネストループをマルチスレッド化するステップとを有することを特徴とする並列プログラム生成方法。

【請求項2】 逐次実行用プログラムを入力して分散共有メモリ型並列計算機用の並列プログラムに変換するコンパイル動作をコンピュータに実行させるための処理手順プログラムを記録するコンピュータ読み取り可能な記録媒体であって、上記コンピュータに、上記逐次実行用プログラムを入力して構文解析を行い中間語を記憶装置に出力する手順と、上記記憶装置から上記中間語を読み出して、ループ繰り返し範囲が変数のループ中に参照を有する配列、もしくは配列宣言寸法が変数の配列、もしくは引数配列を検出する手順と、検出した各配列をページサイズにブロックサイクリック分散させるデータ分散指示文を生成して上記中間語に挿入する手順と、上記データ分散指示文により分散される上記各配列のデータ分散形状と一致するループ分散形状となるループ分散指示文を生成して上記中間語に挿入する手順と、上記中間語における並列化ループを含むネストループをマルチスレッド化する手順とを実行させるためのプログラムを記録したことを特徴とする記録媒体。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】 本発明は、逐次実行用のプログラムから分散共有メモリ型計算機用の並列プログラムを生成する技術に係わり、特に、データ分散によるデータローカルティの最適化を行い高速な並列プログラムを生成するのに好適な並列プログラム生成方法およびその記録媒体に関するものである。

【0002】

【従来の技術】 複数のプロセッサ (CPU: Central Processing Unit) のそれぞれにメモリを設けて構成した分散共有メモリ型並列計算機の論理共有物理分散メモリ (各プロセッサのメモリを論理的に1つのメモリとして取り扱う) を実現する技術の一つとして、論理共有の仮

想メモリ空間をページと呼ばれる単位毎に切り分け、物理的に分散されたメモリに割り付けるものがある。

【0003】 この技術において、どのページをどのプロセッサに割り付けるかを決定する技術として、プログラムにおいて明示的にデータ分散指示文を記述することによりデータ分散形状を指定するデータ分散技術がある。

【0004】 任意の「配列」に関して、そのデータ分散形状を決定する場合、従来の並列プログラム生成技術では、データ分散後のデータ連続長を長くする観点から、なるべく高次元の部分をデータ分散させる方針や、スレッド起動オーバーヘッド削減の観点から、なるべく外側ループを並列化する方針に従って並列化ループを決定してループ並列化指示文を挿入し、並列化ループのループ分散形状と一致するデータ分散形状を決定してデータ分散指示文を挿入していた。

【0005】 以下、このようなデータ分散技術を、廣岡孝志、太田寛、菊池純男著の「分散共有メモリ向け自動データ分散方法の提案」、情報処理学会研究報告、HPC、Vol. 99、No. 66、pp. 101-106 (1999) に述べられている技術に基づき、図11～図13を用いて説明する。

【0006】 図11は、処理対象の逐次実行用ソースプログラムの具体例を示す説明図であり、図12は、図11における逐次実行用ソースプログラムから従来技術により生成された並列プログラムの具体例を示す説明図、図13は、図12における並列プログラムのデータ分散・参照状況例を示す説明図である。

【0007】 図11に示すような逐次実行用ソースプログラム1101が入力された場合、従来の並列化コンパイラを用いた並列プログラム生成システムでは、まず、21行目の外側ループの並列化を決定し、図12に示す並列プログラム1201における13行目の「c\$do」、および、26行目の「c\$end do」に示すようなループ並列化指示文を挿入する。さらに、12行目の「c\$parallel」、および、27行目の「c\$end parallel」に示すようなスレッド起動、終結指示文を挿入する。

【0008】 次に、並列化ループ内の参照状況から配列「A」の2次元目をブロック分散させるデータ分散形状を決定し、図12の3行目に示すようなデータ分散指示文「c\$distribute A(*,block)」を挿入する。

【0009】 例えば、プロセッサ数を4台とし、図11、12における各変数「n1」～「n4」がそれぞれ20、80、20、40、すなわち、「n1」が20、「n2」が80、「n3」が20、「n4」が40の場合、配列A (20, 80) の各要素は、図13 (a) に示すように、「プロセッサ0 (pe0)」に1要素目から400要素目、「プロセッサ1 (pe1)」に401要素目から800要素目、「プロセッサ2 (pe2)」に801要素目から1200要素目、そして、「プロセッサ3 (pe3)」に1201要素目から1600要素

目が割り付けられる。

【0010】尚、図11、図12のソースプログラム中では配列Aを2次元配列で示したが、図13を用いた説明では、簡単のためアドレス上何要素目という表現を用いた。

【0011】この従来技術では、配列宣言寸法が変数の配列、もしくはループ繰返し範囲が変数のループ中に参照を有する配列において、各プロセッサへのデータの割り付けとループ繰返し範囲の割り付けの間に差が生じ、ループの実行においてデータローカリティを悪化させ、並列プログラムの処理速度向上を妨げる可能性があった。

【0012】例えば、図11に示すような逐次実行用ソースプログラム1101が入力された場合で、上述のように、プロセッサ数4台で、図11における変数「n1」が20、変数「n2」が80、「n3」が20、「n4」が40とすると、「配列A」の各要素は、図13(a)に示すように割り付けられるが、ループ中の「配列A」参照範囲は、図13(b)に示すように、「プロセッサ0(pe0)」が1要素目から200要素目、「プロセッサ1(pe1)」が201要素目から400要素目、「プロセッサ2(pe2)」が401要素目から600要素目、そして、「プロセッサ3(pe3)」が601要素目から800要素目となる。

【0013】従って、図13(c)に示すように、「配列A」の201要素目から800要素目が他のプロセッサに割り付けられたデータの参照、すなわちリモート参照になり、ループ中の全参照の75%がリモート参照になってしまう。

【0014】また、引数配列の場合、実プログラムにおいて配列宣言寸法や参照を含むカーネルループのループ繰返し範囲が変数である場合が多く見受けられ、データローカリティに関して上記と同様の結果となる可能性があった。さらに、手続き間でデータ分散情報を引き渡す、もしくは呼び出し先手続きの最初と最後でデータ再分散するなどのオーバーヘッドが費やされていた。

【0015】

【発明が解決しようとする課題】解決しようとする問題点は、従来の技術では、変換対象のプログラムに、ループ繰返し範囲が変数のループ中に参照を有する配列や、配列宣言寸法が変数の配列、もしくは、引数配列がある場合、各プロセッサへのデータの割り付けとループ繰返し範囲の割り付けの間に差が生じてしまう点である。

【0016】本発明の目的は、これら従来技術の課題を解決し、配列宣言寸法が変数の配列や、ループ繰返し範囲が変数のループ中に参照を有する配列、もしくは、引数配列に対して、ループ分散とデータ分散の差を削減する最適なデータ分散を行い、データローカリティを向上させ、並列プログラムの処理速度を高速化させること

を可能とする並列プログラム生成方法およびその記録媒体を提供することである。

【0017】

【課題を解決するための手段】上記目的を達成するため、本発明の並列プログラム生成方法では、データ分散対象配列検出部5aにより、配列宣言寸法が変数の配列や、ループ繰返し範囲が変数ループ中に参照を有する配列、もしくは、引数配列を検出すると、データ分散形状決定部5bにより、この配列をページサイズにブロックサイクリック分散させるデータ分散指示文を生成して挿入し、さらに、データ分散向けループ形状決定部5cにより、このデータ分散形状と一致するループ分散形状となるループ分散指示文を生成して挿入する。そして、並列化ループネストマルチスレッド化部5dにより、並列化ループを含むネストループをマルチスレッド化し、コード生成部6により並列プログラムを生成する。

【0018】

【発明の実施の形態】以下、本発明の実施の形態を、図面により詳細に説明する。図1は、本発明に係わる並列プログラム生成システムの構成例を示すブロック図であり、図2は、図1における並列プログラム生成システムを構成するコンピュータのハードウェア構成例を示すブロック図、図3は、図1における並列プログラム生成システムで生成した並列プログラムを利用する分散共有メモリ型並列計算機システムの構成例を示すブロック図である。

【0019】図3において、31～33はプロセッサ（図中、「CPU」と記載）(1)～(n)、31a～31cは各プロセッサ(1)31～(n)33に個別に物理的に接続されたメモリ(1)～(n)であり、各メモリ(1)31a～(n)31cにより、論理共有メモリ30を構成している。

【0020】このような構成においては、各プロセッサ(1)31～(n)33は、他のプロセッサに接続されたメモリ(1)31a～(n)31cに格納されたデータを利用する場合、すなわち、リモート参照を行う場合、各プロセッサ(1)31～(n)33間での通信処理が必要となり、自プロセッサに接続されたメモリ(1)31a～(n)31cに格納されたデータを利用する場合（ローカル参照）に比べ効率が悪くなる。

【0021】本例では、図1に示す構成の並列化コンパイラを用いて並列プログラムを生成することにより、リモート参照を少なくローカル参照を多くするようにデータ分散できる。すなわち、データローカルティを向上させることができる。

【0022】図2において、21はCRT (Cathode Ray Tube) やLCD (Liquid Crystal Display) 等からなる表示装置、22はキーボードやマウス等からなる入力装置、23はHDD (Hard Disk Drive) 等からなる外部記憶装置、24はCPU (Central Processing Unit)

24 aや主メモリ24 b等からなり蓄積プログラム方式によるコンピュータ処理を行なう情報処理装置、25は本発明の並列プログラム生成方法に係わるプログラムやデータを記録したCD-ROM (Compact Disc-Read Only Memory) もしくはDVD (Digital Video Disc/Digital Versatile Disc) 等からなる光ディスク、26は光ディスク25に記録されたプログラムおよびデータを読み出すための駆動装置、27はLAN (Local Area Network) カードやモデム等からなる通信装置である。

【0023】光ディスク25に格納されたプログラムおよびデータを情報処理装置24により駆動装置26を介して外部記憶装置23内にインストールした後、外部記憶装置23から主メモリ24 bに読み込みCPU24 aで処理することにより、図1に示す並列化コンパイラの各機能が構成される。

【0024】図1において、1は変換対象の対象となる高級言語で記述された逐次実行用のソースプログラム (図中および以下、「逐次プログラム」と記載)、2は逐次プログラム1を入力して、並列実行用の並列プログラムに変換する並列化コンパイラ、3は並列化コンパイラ2で変換され出力される並列プログラムである。

【0025】並列化コンパイラ2は、構文解析部4と、並列化部5、および、コード生成部6を有し、逐次プログラム1を入力として、並列プログラム3を生成する。以下、このような構成の並列化コンパイラ2の処理動作の概要を、図8〜図10を用いて説明する。

【0026】図8は、図1における逐次プログラムの具体例を示す説明図であり、図9は、図1における並列プログラムの具体例を示す説明図、図10は、図9における並列プログラムのデータ分散・参照状況例を示す説明図である。

【0027】図1における並列化コンパイラ2に、図8に示すような逐次プログラム1が入力された場合、例えばページサイズを配列5要素分とすると、まず、図9の3行目に示すように、「配列A」を、「ページサイズ5」でブロックサイクリックにデータ分散させるためのデータ分散指示文「c\$distributed A(cyclic(5),*)」を挿入する。尚、図8における逐次プログラム1は、従来技術での説明で用いた図11における逐次実行用プログラム1101と同じ内容である。

【0028】次に、このデータ分散形状と一致するループ分散形状となるループ分散指示文「c\$do schedule(static,5)」を図9の22行目に、また、並列化ループの終端を示す指示文「c\$end do」を26行目に挿入する。そして、並列化の対象となるループネストをマルチスレッド化するため、図9の11行目の「c\$parallel」、および、28行目の「c\$end parallel」に示すスレッド起動、終結指示文を挿入する。

【0029】この図9に示す並列プログラム3を実行することにより、「配列A」は、図10(a)に示すよう

に、ページサイズと等しい「サイズ5」でサイクリックに各プロセッサ(「プロセッサ0」〜「プロセッサ3」)に割り付けられる。また、ループにおける参照範囲も、図10(b)に示すように、「サイズ5」でサイクリックに各「プロセッサ0」〜「プロセッサ3」に割り付けられるので、データ分散と処理分散の差がなくなり、図10(c)に示すように、100%ローカル参照となる。

【0030】次に、図1を用いて、構文解析部4と並列化部5およびコード生成部6を有する並列化コンパイラ2の詳細を説明する。並列化コンパイラ2は、逐次プログラム1を入力し、並列実行用の並列プログラム3を生成、出力し、その処理の過程で中間語8を生成する。尚、並列化コンパイラ2の出力である並列プログラム3は、本例ではソースプログラム形式で示すが、一般にはソースプログラム形式とは限らない。

【0031】並列化コンパイラ2の構文解析部4は、逐次プログラム1を読み込み、構文解析を行って中間語8を生成する。この中間語8は、図2における外部記憶装置23または主メモリ24 bの記録領域に記録される。また、並列化部5は、この中間語8から逐次プログラム1中のデータ分散の対象となる配列を検出し、データ分散指示文を生成し、さらにループ分散指示文を生成し、そして並列化ループを含むネストループをマルチスレッド化して中間語8を複数個のプロセッサで並列に処理する構造を持った中間語8に変換する。そして、コード生成部6は、並列化部5で変換された中間語8から並列プログラム3を生成して出力する。

【0032】並列化部5は、データ分散対象配列検出部5aと、データ分散形状決定部5b、データ分散向けループ分散形状決定部5c、および、並列化ループネストマルチスレッド化部5dを有している。

【0033】データ分散対象配列検出部5aは、入力した逐次プログラム1中のループの繰り返し範囲が変数のループ中に参照を有する配列、もしくは配列宣言文が変数の配列、もしくは引数配列を検出する。また、データ分散形状決定部5bは、データ分散対象配列検出部5aで検出した配列をページサイズにブロックサイクリック分散させるデータ分散指示文を生成して挿入する。

【0034】さらに、ループ分散形状決定部6により、データ分散形状決定部5bによるデータ分散形状と一致するループ分散形状となるループ分散指示文を生成して挿入する。そして、並列化ループネストマルチスレッド化部5dにより、並列化ループを含むネストループをマルチスレッド化する。

【0035】次に、このような構成の並列化部5による並列プログラム生成動作を、図4から図7を用いて説明する。

【0036】図4は、本発明に係わる並列プログラム生成方法の処理手順例を示すフローチャートであり、図5

は、図1におけるデータ分散対象配列検出部の処理動作例を示すフローチャート、図6は、図1における解析情報（ループテーブル）の具体例を示す説明図、図7は、図1における解析情報（配列テーブル）の具体例を示す説明図である。

【0037】ここでは、図8に示す逐次プログラム1が入力された場合を例に説明する。図1の並列化部5では、まず、データ分散対象配列検出部5aにより、入力した逐次プログラム1中のループ繰り返し範囲が変数のループ中に参照を有する配列、もしくは配列宣言文法が変数の配列、もしくは引数配列を検出する（ステップ41）。

【0038】この図1におけるデータ分散対象配列検出部5aによる各配列の検出動作の詳細が、図5に示すフローチャートであり、このデータ分散対象配列検出の際に作成される図1における解析情報7としてのループテーブルの例が図6、配列テーブルの例が図7に示されている。これらのループテーブルと配列テーブルからなる解析情報7は、図2における外部記憶装置23や主メモリ24bの記憶領域に記録される。

【0039】以下、図5を用いて、図4に示すステップ41におけるデータ分散対象配列検出処理を説明する。まず、ステップ411の処理において、図8の逐次プログラム1中の21行目から25行目までの並列化可能ループを検出する。次に、ステップ412の処理において、図6のループテーブル811における第1番目のループテーブル812を生成する。

【0040】図6におけるループテーブル811には、テーブル番号911、NEXTテーブル912、ループポインタ913、ループ制御変数（図中、「LCV」と記載）914、ループ制御変数の下限式（図中、「LCVの下限式」と記載）915、ループ制御変数の上限式（図中、「LCVの上限式」と記載）916、ループ制御変数の増分式（図中、「LCVの増分式」と記載）917、並列化ループフラグ918の各項目に対応する情報が含まれる。

【0041】本例では、第1番目のループテーブル812の各項目の値として、テーブル番号911に「1」、NEXTテーブル912は未登録、ループポインタ913に「21」（図8の21行目を参照）、ループ制御変数914に「i」、ループ制御変数の下限式915に「1」、ループ制御変数の上限式916に「n4」、ループ制御変数の増分式917に「1」、並列化ループフラグ918に（最初の動作では）「false」が設定される。

【0042】次に、ステップ413の処理において、最内ループであるか否かを判定する。この時点では、最内ループでないのでステップ415の処理を行う。このステップ415の処理では、他のループが存在するか否かを判定する。ここでは、他にループが存在するので、ス

テップ411の処理に戻り、各ステップでの処理を繰り返す。

【0043】今回は、ステップ411の処理で、図8の逐次プログラム1中の22行目から24行目までの並列化可能ループを検出する。そして、ステップ412の処理で、図6に示すような第2番目のループテーブル813を生成する。

【0044】本例では、第2番目のループテーブル813の各項目の値として、テーブル番号911に「2」、NEXTテーブル912は未登録、ループポインタ913に「22」（図8の22行目参照）、ループ制御変数914に「j」、ループ制御変数の下限式915に「1」、ループ制御変数の上限式916に「n3」、ループ制御変数の増分式917に「1」、並列化ループフラグ918に「false」を設定し、第1番目のループテーブル812のNEXTテーブル912に「2」を設定する。

【0045】そして、ステップ413の処理で、最内ループであるか否かを判定し、今回は最内ループであるため、ステップ414の処理を行う。このステップ414の処理では、図6に示す第2番目のループテーブル813における並列化ループフラグ918に「true」を設定する。

【0046】さらに、ステップ415の処理で、他のループが存在するか否かを判定し、ここでは、他にループが存在しないため、ステップ416の処理を行う。このステップ416の処理では、並列化ループ中に参照を有してデータ分散の対象となるターゲット配列の候補として「配列A」を検出する。

【0047】次のステップ417の処理では、図7に示す配列テーブル821における第1番目の配列テーブル822を生成する。この第1番目の配列テーブル822には、テーブル番号921、NEXTテーブル922、配列名923、次元数924、1次元目の宣言文法925、2次元目の宣言文法926、引数フラグ927、ターゲット配列フラグ928の各項目に対応する情報が含まれる。

【0048】本例では、第1番目の配列テーブル822の各項目の値として、テーブル番号921に「1」、NEXTテーブル922は未登録、配列名923に「A」、次元数924に「2」、1次元目の宣言文法925に「1:n1」、2次元目の宣言文法926に「1:n2」、引数フラグ927に「true」、ターゲット配列フラグ928に（最初の動作で）「false」が設定される。

【0049】ステップ418の処理で、他の並列化ループが存在するか否かを判定し、ここでは他に並列化ループが存在しないため、ステップ419の処理を行う。このステップ419の処理では、「配列A」がターゲット配列であるか否かの判定を行う。本例では、「配列A」

が引数配列であり、また、宣言寸法が変数であるため、ターゲット配列と判定し、ステップ420の処理を行う。

【0050】このステップ420の処理では、図7に示す第1番目の配列テーブル822のターゲット配列フラグ928に「true」を設定する。その後、ステップ421の処理で、他の配列テーブルが存在するかどうかを判定し、ここでは他に配列テーブルが存在しないため、本データ分散対象配列検出処理を終了する。

【0051】このようなターゲット配列の検出後は、図4におけるステップ42からの処理を行う。すなわち、図4におけるステップ42の処理では、ターゲット配列Aに対し、ページサイズにブロックサイクリック分散させるデータ分散指示文を生成し挿入する。例えば、ページサイズを配列要素5要素分とした場合、データ分散指示文「c\$distribe A(cyclic(5),*)」を生成し挿入する。尚、このステップ42での処理は、図1のデータ分散形状決定部5bで実行される。

【0052】次に、ステップ43の処理では、データ分散形状決定部5bでのデータ分散形状と一致するループ分散形状となるループ分散指示文「c\$do schedule(static,5)」を生成し、図9に示すプログラムにおける22行目に挿入する。また、26行目に並列化ループの終端を示す指示文「c\$end do」を挿入する。尚、このステップ43の処理は、図1のデータ分散向けループ分散形状決定部5cで実行される。

【0053】このデータ分散向けループ分散形状決定部5cによる処理後、ステップ44の処理を行い、並列化ループを含むネストループをマルチスレッド化する。具体的には、図9に示す11行目における指示文「c\$parallel」、および28行目における指示文「c\$end parallel」を挿入する。尚、このステップ44での処理は、図1の並列化ループネストマルチスレッド化部5dで実行される。

【0054】以上、図1～図10を用いて説明したように、本例の並列プログラム生成システムと方法では、ループ繰り返し範囲が変数のループ中に参照を有する配列、もしくは配列宣言寸法が変数の配列、もしくは引数配列を検出し、それぞれの配列に対し、ページサイズでブロックサイクリックデータ分散させるデータ分散指示文を生成して挿入し、さらに、このデータ分散形状と一致するループ分散形状となるループ分散指示文を生成して挿入し、そして、並列化ループを含むネストループをマルチスレッド化して、並列プログラムを生成することにより、各プロセッサへのデータの割り付けとループ中における参照範囲を一致させることができる。このことにより、データローカリティが向上し、並列プログラムの処理速度が高速化される。

【0055】尚、本発明は、図1～図10を用いて説明した例に限定されるものではなく、その要旨を逸脱しな

い範囲において種々変更可能である。例えば、本例では、CD-ROMやDVD等からなる光ディスクを本発明の並列プログラム生成方法に係わる処理プログラムやデータを記録する記録媒体として用いているが、FD (Flexible Disk) 等、他の記録媒体を用いることでも良い。また、プログラムのインストールに関しても、記録媒体ではなく、例えばモデムやターミナルアダプタ(TA)等からなる通信装置を介してネットワークからダウンロードして外部記憶装置にインストールすることでも良い。

【0056】

【発明の効果】本発明によれば、変換対象のプログラムに、ループ繰り返し範囲が変数のループ中に参照を有する配列や、配列宣言寸法が変数の配列、もしくは、引数配列がある場合であっても、各プロセッサへのデータの割り付け(データ分散)とループ繰り返し範囲の割り付け(ループ分散)の間の差を削減する最適なデータ分散を行うことができ、データローカリティが向上し、並列プログラムの処理速度を高速化させることが可能である。

【図面の簡単な説明】

【図1】本発明に係わる並列プログラム生成システムの構成例を示すブロック図である。

【図2】図1における並列プログラム生成システムを構成するコンピュータのハードウェア構成例を示すブロック図である。

【図3】図1における並列プログラム生成システムで生成した並列プログラムを利用する分散共有メモリ型並列計算機システムの構成例を示すブロック図である。

【図4】本発明に係わる並列プログラム生成方法の処理手順例を示すフローチャートである。

【図5】図1におけるデータ分散対象配列検出部の処理動作例を示すフローチャートである。

【図6】図1における解析情報(ループテーブル)の具体例を示す説明図である。

【図7】図1における解析情報(配列テーブル)の具体例を示す説明図である。

【図8】図1における逐次プログラムの具体例を示す説明図である。

【図9】図1における並列プログラムの具体例を示す説明図である。

【図10】図9における並列プログラムのデータ分散・参照状況例を示す説明図である。

【図11】処理対象の逐次実行用ソースプログラムの具体例を示す説明図である。

【図12】図11における逐次実行用ソースプログラムから従来技術により生成された並列プログラムの具体例を示す説明図である。

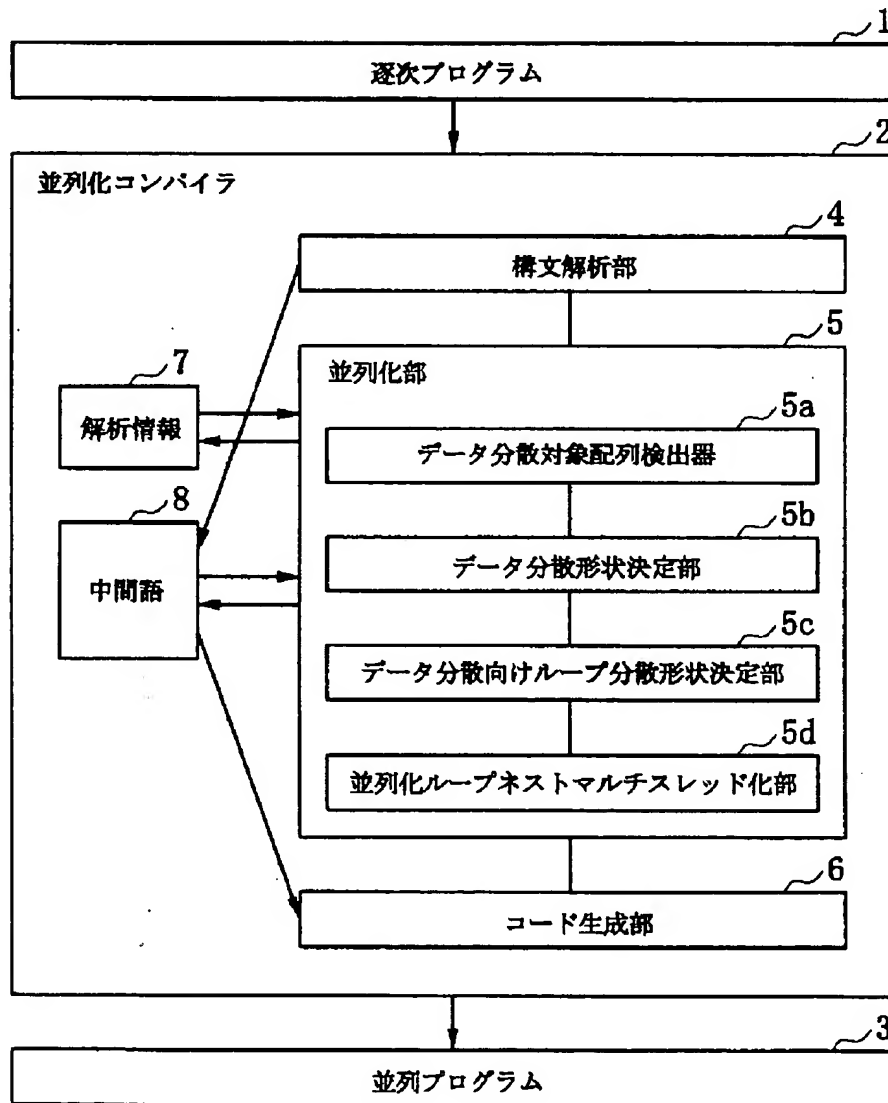
【図13】図12における並列プログラムのデータ分散・参照状況例を示す説明図である。

【符号の説明】

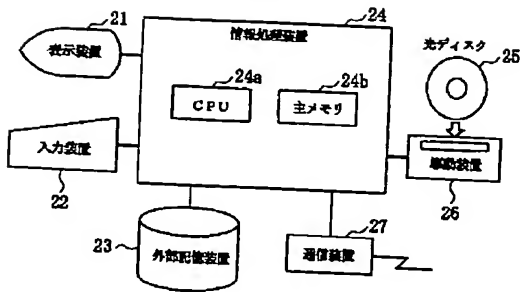
1：逐次プログラム、2：並列化コンパイラ、3：並列プログラム、4：構文解析部、5：並列化部、5a：データ分散対象配列検出部、5b：データ分散形状決定部、5c：データ分散向けループ分散形状決定部、5d：並列化ループネストマルチスレッド化部、6：コード生成部、7：解析情報、8：中間語、21：表示装置、22：入力装置、23：外部記憶装置、24：情報処理装置、25：光ディスク、26：駆動装置、27：通信装置、30：論理共有メモリ、31～33：プロセッサ（「CPU」）、31a～33a：メモリ、811：ループテーブル、812：第1番目のループテーブル、813：第2番目のループテーブル、821：配列

テーブル、822：第1番目の配列テーブル、911：テーブル番号、912：NEXTテーブル、913：ループポインタ、914：ループ制御変数（「LCV」）、915：ループ制御変数の下限式（「LCVの下限式」）、916：ループ制御変数の上限式（「LCVの上限式」）、917：ループ制御変数の増分式（「LCVの増分式」）、918：並列化ループフラグ、921：テーブル番号、922：NEXTテーブル、923：配列名、924：次元数、925：1次元目の宣言寸法、926：2次元目の宣言寸法、927：引数フラグ、928：ターゲット配列フラグ、1101：逐次実行用ソースプログラム、1201：並列プログラム。

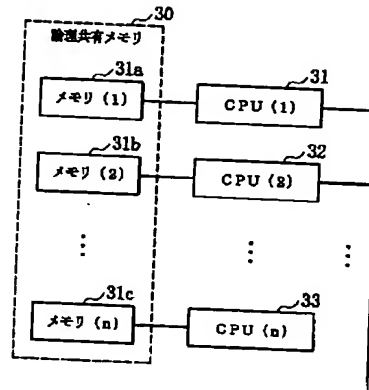
【図1】



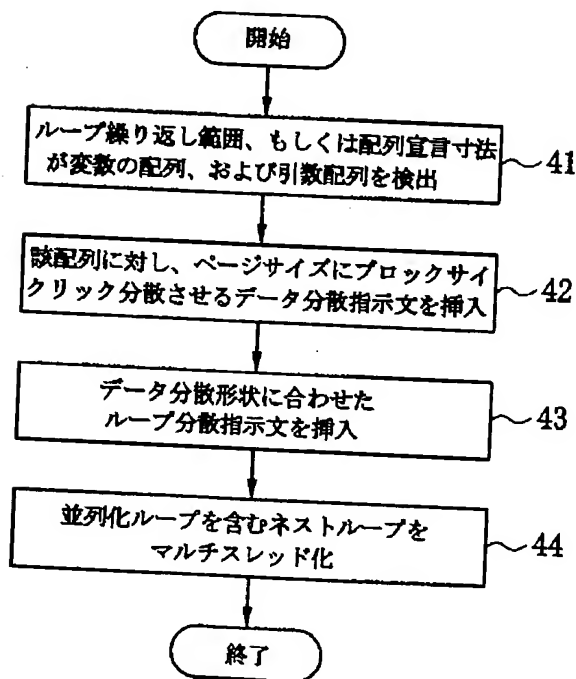
【図2】



【図3】



【図4】



【図8】

```

1: subroutine sub1 (A, n1, n2)
2: real A(n1, n2)
.....
21: do i=1, n4
22:   do j=1, n3
23:     A(j, i)=...
24:   enddo
25: enddo
26: return

```

【図6】

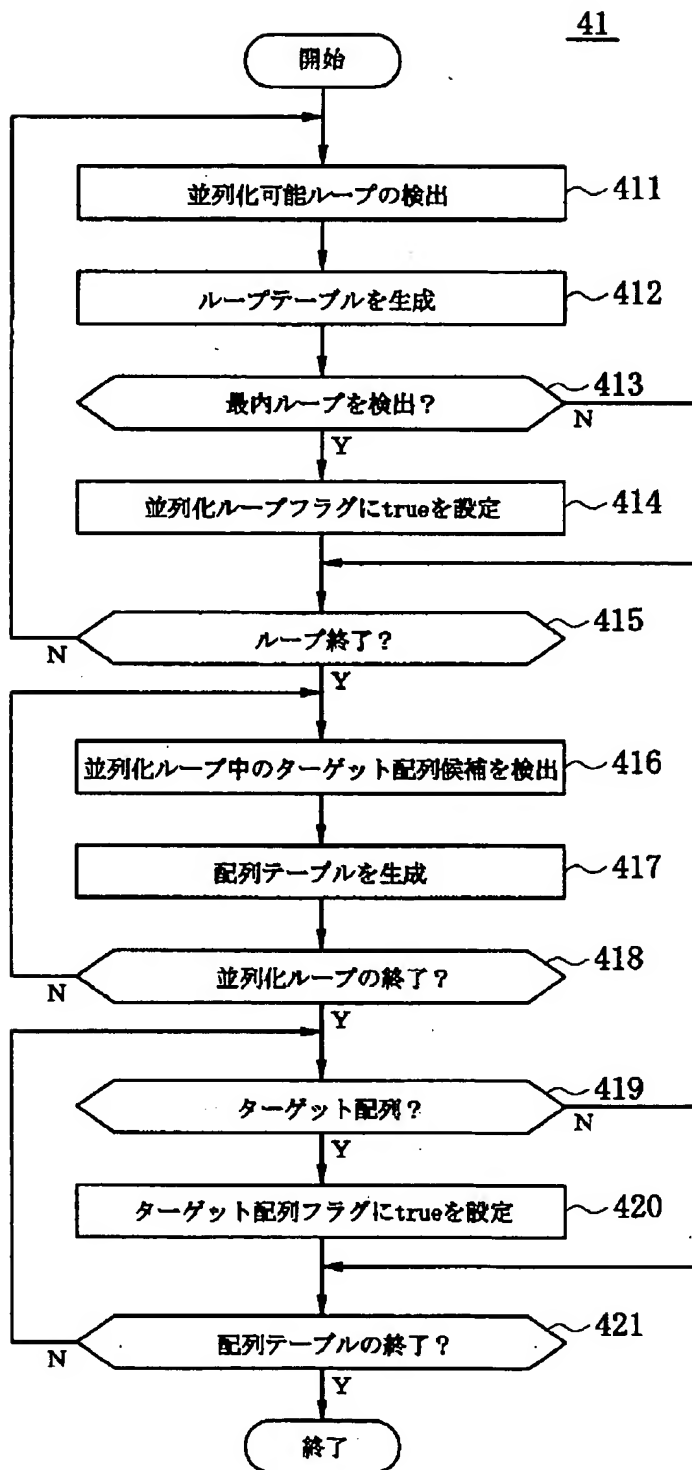
ループテーブル		
	812	813
911~ テーブル番号	1	2
912~ NEXTテーブル	2	-
913~ ループポイント	2 1	2 2
914~ LCV	1	i
915~ LCVの下限式	1	1
916~ LCVの上限式	n 4	n 3
917~ LCVの増分式	1	1
918~ 並列化ループフラグ	false	true

LCV: ループ制御変数

【図7】

配列テーブル	
	822
921~ テーブル番号	1
922~ NEXTテーブル	-
923~ 配列名	A
924~ 次元数	2
925~ 1次元目の宣言寸法	1:n 1
926~ 2次元目の宣言寸法	1:n 2
927~ 引数フラグ	true
928~ ターゲット配列フラグ	true

【図5】



【図9】

3

```

1: subroutine sub1(A,n1,n2)
2: real A(n1,n2)
3: c$distributed A(ayolio(5),*)
.....
11: c$parallel
21: do i=1, n4
22: c$do schedule(static,5)
23: do j=1, n3
24:   A(j,i)=...
25: enddo
26: c$end do
27: c$enddo
28: c$end parallel
29: return
  
```

【図11】

1101

```

1: subroutine sub1(A,n1,n2)
2: real A(n1,n2)
.....
21: do i=1, n4
22:   do j=1, n3
23:     A(j,i)=...
24:   enddo
25: enddo
26: return
  
```

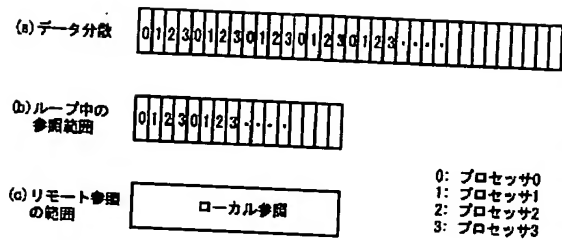
【図12】

1201

```

1: subroutine sub1(A,n1,n2)
2: real A(n1,n2)
3: c$distributed A(w,block)
.....
12: c$parallel
13: c$do
21: do i=1, n4
22:   do j=1, n3
23:     A(j,i)=...
24:   enddo
25: enddo
26: c$end do
27: c$end parallel
28: return
  
```

【図10】



【図13】

